Dual Averaging with Adaptive Random Projection for Solving Evolving Distributed Optimization Problems

Shreyas Vathul Subramanian, Daniel A. DeLaurentis & Dengfeng Sun

Journal of Optimization Theory and Applications

ISSN 0022-3239
Volume 170
Number 2

J Optim Theory Appl (2016) 170:493-511
DOI 10.1007/s10957-016-0932-z

J Optim Theory Appl (2016) 170:493–511
DOI 10.1007/s10957-016-0932-z

Dual Averaging with Adaptive Random Projection for Solving Evolving Distributed Optimization Problems

Shreyas Vathul Subramanian[1] · Daniel A. DeLaurentis[1] · Dengfeng Sun[1]

Received: 18 April 2015 / Accepted: 26 March 2016 / Published online: 7 April 2016

Abstract We study a sequential form of the distributed dual averaging algorithm that minimizes the sum of convex functions in a special case where the number of functions increases gradually. This is done by introducing an intermediate 'pivot' stage posed as a convex feasibility problem that minimizes average constraint violation with respect to a family of convex sets. Under this approach, we introduce a version of the minimum sum optimization problem that incorporates an evolving design space. Proof of mathematical convergence of the algorithm is complemented by an application problem that involves finding the location of a noisy, mobile source using an evolving wireless sensor network. Results obtained confirm that the new designs in the evolved design space are superior to the ones found in the original design space due to the unique path followed to reach the optimum.

Keywords Convex optimization · Dual averaging · Maximum feasibility problem · Distributed · Sensor management · Topology optimization

Mathematics Subject Classification 90C25 · 90C30 · 90C46

✉ Shreyas Vathul Subramanian
subram14@purdue.edu

Daniel A. DeLaurentis
ddelaure@purdue.edu

Dengfeng Sun
dsun@purdue.edu

[1] Purdue University, West Lafayette, IN, USA

1 Introduction

Several categories of optimization problems and their resulting applications focus on optimizing a sum of objective functions. Some examples include least squares and inference, stochastic programming, dual optimization and machine learning (as seen in [1,2]). Bertsekas [2] provides a unified framework along with convergence and convergence rates of a variety of methods that can be derived from three basic solution strategies—incremental gradient, incremental subgradient and proximal methods. However, all the methods and variants presented in the literature only tackle a problem of fixed size (or number of nodes, n). Although Nedic [3] use a subgradient method for solving a distributed optimization problem over time-varying graphs, the number of vertices remains fixed. The method we propose will also be effective in situations where interaction between nodes needs to be considered. To include these interaction effects, 'link' functions may be used in addition to 'vertex' functions in the sum of objective functions form. Another situation worth discussing is the use of purely local functions, that is, f_i's that are only functions of local variables, x_i's. In this situation, a distributed optimization problem may be formulated through the addition of some form of interaction (therefore avoiding the trivial result of a sum of local optimum function values for the entire optimum of a sum of functions). We present a novel method that combines distributed dual averaging (Sect. 3.1) and the random Bregman projection method (Sect. 3.2) that solves our version of the time-varying minimum sum problem (Sect. 2).

2 Problem Description

We are interested in convex optimization problems of the form shown in (1). Define the problem at time t, $\mathscr{P}_t$ as follows

$$\mathscr{P}_t := \min_x \sum_{i=1}^{n} f_i(x), \quad x \in X, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function (may not be differentiable) and X is a non-empty, closed and convex set with $X \subseteq \mathbb{R}^n$. In our case, the optimization problem at time t ($\mathscr{P}_t$) may evolve at a certain time (say $t+1$) to include p more individual functions. For example, problem $\mathscr{P}_{t+1}$ may be given as:

$$\mathscr{P}_{t+1} := \min_x \sum_{i=1}^{n+p} f_i(x), \quad x \in X \tag{2}$$

with $X \in \mathbb{R}^{n+p}$. We consider an equivalent optimization problem in (3) which is based on functions that are distributed over a network specified by an undirected graph at time t, $G_t = (V_t, E_t)$ over the vertex set $V_t \subseteq \{1, 2, \ldots, n\}$ with edge set $E_t \subseteq V_t \times V_t$.

$$\min_x \sum_{i \in V_t} f_i(x), \quad x \in X \tag{3}$$

To relate (2) and (3), we note that adding a vertex to the network is akin to adding a set of p variables to the optimization problem. Each vertex is associated with an agent that calculates a local function f_i, typically a function of p variable dimensions of the total n dimensions in $x \in X$, with $X \subseteq \mathbb{R}^n$.

Define f as follows—$f := \sum_{i \in V_t} f_i(x) : x \in X$. Let $f_t^* = f(x) : x_t^* \in X$ be the solution of the problem $\mathscr{P}_t$. Furthermore, define set $t+ := [t' : t' > t]$ and t_{end} as the context-specific final time. We discuss our answers to the following questions through this paper:

1. Given an optimization problem ($\mathscr{P}_0$) of the form in (3) that is associated with a graph G_0 at time $t = 0$, how do we sequentially obtain better f_{t+}^* as new vertices are included in the graph G_{t+}? This also defines our motivation—to find a minimum such that $f_{t+}^* < f_t^*$.
2. Is there any benefit of finding the trace of f_t^* sequentially rather than solving a static problem at t_{end}?
3. When does one stop considering the option of changing the number of vertices $n(V_t)$ in graph G_t to obtain a final $f_{t_{\text{end}}}^*$?

3 Foundations of the Algorithm

The goal of decentralized or distributed optimization is to optimize a ‘global’ objective using only ‘local’ (or node specific) function evaluations. Implementation of these optimization algorithms to tracking and localization problems, sensor networks and multi-agent coordination (see [4,5]) have proved to be very effective in the past. Although the form of the function handled by Duchi et al. [6] is slightly different, our work is an offshoot of the method presented therein and repurposes the distributed dual averaging algorithm discussed in Sect. 3.1 to effectively handle an evolving design space in Sect. 3.2.

3.1 Distributed Dual Averaging

3.1.1 Basic Assumptions

Standard dual averaging schemes are based on a proximal function $\phi : X \to \mathbb{R}$ that is strongly convex with respect to some norm $\|\cdot\|$. In our paper, we use the canonical form of this proximal function, $\phi(x) = 1/2\|x\|_2^2$ which is strongly convex with respect to the L_2 norm for $x \in \mathbb{R}^n$. Also, we assume that all nodal cost functions (f_i's) are L-Lipchitz with respect to each norm as in Duchi et al. [6] That is,

$$|f_i(x) - f_i(y)| \leq L\|x - y\|, \ \forall x, y \in X \tag{4}$$

The functions used in our paper are all convex functions in compact domains. An important inference of (4), used in the proof of convergence of the distributed dual averaging (DDA) algorithm, is that for any $x \in X$ and any subgradient $g_i \in \partial f_i(x)$, $\|g\|_* \leq L$.[1] Define the adjacency matrix $A_t \in \mathbb{R}^{n\times n}$ of graph G_t as follows:

$$A_t(i,j) = \begin{cases} 1, & \text{iff } (i,j) \in E_t, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Each node i in the network is associated with neighbors j, denoted as $j \in N(i) := \{j \in V_t : (i,j) \in E_t\}$. Let degree δ_i of each node i be defined as $\delta_i := |N(i)| = \sum_{j=1}^{n} A_t(i,j)$, and $D = diag\{\delta_1, \ldots, \delta_n\}$. Let $\delta_{\max} = \max_{i\in V_t} \delta_i$, and let us define the characteristic matrix P as:

$$P(G_t) := I - \frac{1}{\delta_{\max}+1}(D - A) \tag{6}$$

The matrix P is symmetric and doubly stochastic ($\sum_i P_{ij} = \sum_j P_{ij} = 1$) by construction; these are characteristics needed by Duchi et al. [6] for their convergence proof to be valid, which is an essential part of our algorithm.

3.1.2 Algorithm Details

First we note that the distributed dual averaging is implemented at a particular fixed time (t). The details described here are for iterations (counted using the parameter $k = 1, 2, \ldots$) that occur at a time t during the evolution of the problem to $\mathscr{P}_{t+}$. During these iterations, each node maintains a local copy of two values $\{x_i(k), z_i(k)\} \in X \times \mathbb{R}$ and also calculates the subgradient $g_i(k)$, where $g_i(k) \in \partial f_i(x_i(k))$. Each node receives information about parameters in the form of $\{z_j(k), j \in N(i)\}$ from nodes in its neighborhood. Define a projection onto X with respect to some proximal function ϕ and positive scalar step-size α as

$$\Pi_X^{\phi}(z,\alpha) := \arg\min_{x\in X} \left\{ < z, x > + \frac{1}{\alpha}\phi(x) \right\} \tag{7}$$

Given a non-increasing sequence of positive step sizes $\alpha(k)$, the DDA algorithm involves each node $i \in V_t$ via the following iterative updates at a fixed time t:

$$\begin{aligned} z_i(k+1) &= \sum_{j\in N(i)} P_{ji} z_j(k) + g_i(k) \\ x_i(k+1) &= \Pi_X^{\phi}(z_i(k+1), \alpha(k)) \end{aligned} \tag{8}$$

We refer the reader to [6] for convergence of the DDA algorithm to the optimum $x^* \in X$, convergence rates and modifications for various network types (random, fixed degree, etc.). The method described in [6] is only suited to solve static problems and not the dynamic or evolving problems of interest to us. Modification of this algorithm to handle evolving design spaces is supported by 1) strong convergence

[1] $\|\cdot\|_*$ is the dual norm to $\|\cdot\|$. See [6] for details.

results for variants of DDA that include stochastic communication links and composite objective function forms; and 2) superior experimental performance when compared to state-of-the-art algorithms such as Markov incremental gradient descent (MIGD)[7] and distributed projected gradient method [8]. The following section introduces the Bregman projection which is helpful in motivating the need for our adaptive random projection (ARP) algorithm in the pivot phase.

3.2 Bregman Projection Method

Of the several algorithms related to steered sequential projections described by Censor et al. [9], we are interested in algorithms that use Bregman distance functions for projection onto a set. Bregman distance is defined on a convex closed set $S \subset \mathbb{R}^n$ with respect to a function f. A Bregman distance is denoted by $D_f(x, y) : S \times \text{int}(S) \to \mathbb{R}$ for the distance between two points $x \in S$ and $y \in \text{int}(S)$ as:[2]

$$D_f(x, y) := f(x) - f(y) - < \nabla f(y), x - y > \tag{9}$$

If $\Omega \subseteq \mathbb{R}^n$ is closed and convex, and $\Omega \cap S \neq \emptyset$ a Bregman projection onto the set Ω (denoted as Π^f_Ω) is then defined as:

$$\Pi^f_\Omega(x) = \arg\min_z \{D_f(z, x) : z \in \Omega \cap S\} \tag{10}$$

where uniqueness of the projection $\Pi^f_\Omega(x)$ is ensured for $z \in \text{int}(S)$ for Bregman functions f. More information about Bregman projections and their use in convex optimization may be obtained from [10] and [11]. We use a special, quadratic form of the Bregman function, $1/2\|x\|^2_2$ (with zone $\mathbb{R}^n$). Recall that this Bregman function has the same form as the proximal function used in Sect. 3.1. In this case, Bregman projections become orthogonal (Euclidian) projections. The iterates of the algorithm are defined as

$$x(k+1) = x(k) + \sigma_k(\Pi_{i(k)}(x(k)) - x(k)) \tag{11}$$

where $\{i(k)\}$ is a cyclic control sequence, and the sequence $\{\sigma_k\}_{k\geq 0}$ is an m-steering sequence as defined in Censor et al. [9], and $\Pi_{i(k)}$ are successive projection operators. However, although the steered version of the Bregman Projection method can solve inconsistent problems, it cannot be applied to evolving optimization problems where $\mathscr{P}_{t-} \neq \mathscr{P}_t \neq \mathscr{P}_{t+}$.

Given that we are able to find the optimum f^*_t of a problem $\mathscr{P}_t$ of the form in (3) through updates of the DDA algorithm (8) at some time $t > 0$, our aim is to find a superior $f^*_{t+} < f^*_t$ by activating or deactivating certain nodes in the network ($i \in V_{t+}, V_{t+} \subseteq \{1, 2, \ldots, n\}, V_{t+} \neq V_t$). In other words, we are modifying the network by selecting a new set of nodes from a 'library' of n existing nodes. Solving the problem $\mathscr{P}_{t+}$ using DDA again may find an $f^*_{t+} < f^*_t$, but we are ignoring the progress made through solving the problems $\mathscr{P}_0$ to $\mathscr{P}_t$. Thus, a form of numerical

[2] With int(S) as the interior of the closed convex set S.

continuation is required and is established through a 'pivot' phase at time t, before solving the problem $\mathscr{P}_{t+}$.

At time t, problem $\mathscr{P}_t$ in (3) is solved to obtain an optimum f^* and a corresponding value of $x^* \in X$. Let $\triangle V_t$ be the symmetric difference between the sets V_t and V_{t+}.[3] This gives us the nodes added or removed in the new time step $t+$. Typically, nodes are added to the network, as seen in our application problem in the penultimate section. Thus, we can rewrite (3) as follows

$$\min_x \sum_{i \in V_{t+}} f_i(x) = \min_x \left(\sum_{i \in V_t} f_i(x) \pm \sum_{i \in \triangle V_t} f_i(x) \right), \quad x \in X \tag{12}$$

Since the first sum of functions has been solved for in the previous time step with $i \in V_t$, the problem at the pivot step is reduced to a simpler, convex feasibility problem. Since we wish to find a *better* objective function value f, a point x at which $f < f^*$ is desired. Also, an alternative design point (x) at a different location, but with the same optimum function value $(f = f^*)$, may also be valuable. Since the functions f_i are convex, the sum of these functions is also convex, with f^* being a scalar value that defines a specific boundary or contour line of $f \leq f^*$ (which itself is convex in nature). f^* now represents a minimum acceptable performance threshold. Given our new, but related goal of finding a better f^*, this convex feasibility problem can be written as:

$$\min_x \; < \mathbf{0}, x >, \; f^*_{t+1} = \sum_{i \in V_{t+}} f_i(x) < f^*_t \text{ and } x \in X \tag{13}$$

Represent the convex set $x \in X$ at any pivot time as Q_X, and the set $\sum_{i \in V_{t+}} f_i(x) < f^*_t$ as Q_F. Thus, at the pivot time, our solution is a point in the intersection of these sets, that is, $\tilde{x} \in Q_X \cap Q_F$. There exist several convex feasibility algorithms for finding a point $x^* \in Q := \cap_{i=0}^{m-1} Q_i$, which is the intersection of finitely many closed, individual sets $Q_i \in \mathbb{R}^n$ [12–14]. However, most of these algorithms are applicable only to the 'consistent' case wherein $Q \neq \emptyset$. In our case, it is not known a priori that $Q \neq \emptyset$ at time $t+$. Thus, a sequential projection method that is proved to be convergent for the inconsistent as well as consistent case is used here [9]. Here, we are interested in an iterative algorithm to find a point in the intersection of the sets Q_X and Q_F. Additionally, as in [3], we would also like to determine whether such a point exists by observing the algorithm's iterates itself. Our adaptive random projections (ARP) algorithm (introduced and analyzed in the following sections) is well suited to solve such dynamic problems.

4 Adaptive Random Projection

Our algorithm is based on the hypothesis that finding a point in the intersection of Q_i's corresponds to an improvement in f^*. We prove this hypothesis using the fol-

[3] Symmetric difference $\triangle V_t = V_t \triangle V_{t+} = (V_t \cup V_{t+}) \setminus (V_t \cap V_{t+})$.

lowing argument. We do not provide this explanation as a formal proof, but rather as a motivating yet necessary discussion to lead to the development of our algorithm.

Consider an existing $f^* \geq \sum_i f_i(x) = F(x)$ and a pair of new candidate functions f_1 and f_2. First, suppose that adding f_1 does not satisfy the feasibility criterion in (13) and that adding f_2 does satisfy it. In other words, (a) $F(x) + f_1(x) - f^* > 0 \, \forall x \in Q_1$ but (b) $F(x) + f_2(x) - f^* \leq 0 \, \forall x \in Q_2$. We are interested in the case where both these conditions (a) and (b) are simultaneously true, that is, $Q_1 \cap Q_2 \neq \emptyset$. Exclusion of the node corresponding to f_1 is justified since $f^* \geq F \; \forall x \in Q_1$, and $F(x) + f_1(x) + f_2(x) - f^* \leq f_1(x) \; \forall x \in Q_1 \cap Q_2$. Of course, when $Q_1 \cap Q_2 = \emptyset$, we cannot find a common point x to evaluate f_1 and f_2.

Now consider another candidate function f_3 with (c) $F(x)+f_3(x)-f^* \leq 0 \; \forall x \in Q_3$ being true. We are interested in the scenario where (b) and (c) are simultaneously true. Observe that when $Q_2 \cap Q_3 \neq \emptyset$, $F(x) + f_2(x) + f_3(x) - f^* \leq f_2(x)$ and $F(x) + f_2(x) + f_3(x) - f^* \leq f_3(x)$ are both true. Since $f_2 \leq f^* - F(x)$ and $f_3 \leq f^* - F(x)$, the expression $F(x)+f_2(x)+f_3(x)-f^* \leq 0$ always holds true. Also, there may be a greater benefit when both functions f_2 and f_3 are considered simultaneously than with any one of the functions alone. More nodes with corresponding functions f_i can be added when (b) and (c) are found to be true. With this motivation, we now proceed to introduce our algorithm.

4.1 Algorithm Details

Section 3.2 introduces the problem at hand as a convex feasibility problem. More precisely, our problem is related to a body of research on the maximum feasible subset (MFS) problem in which the goal is to identify (enumerate) the sets that have a feasible intersection. In view of our overall goal, a feasible intersection of a subset of the candidate library of nodes translates to a definite improvement of the global objective function by adding these nodes. Other closely related problems include the minimum unsatisfied linear relation problem (MINULR) and minimum cardinality set covering problem (MINCOVER) that are typically applied to a set of linear constraints. Finding the MFS of linear constraints is NP-hard and is widely studied (see [15–17]).

In the pivot stage, we find a point x^* such that it lies in the intersection of a select subset of $x^* \in X$ with $X \triangleq \bigcap_{i \in \mathscr{I}} X_i$, where $\mathscr{I} \in \{1, \ldots, m\}$, $|\mathscr{I}| \geq 2$. Our proposed adaptive random projection algorithm is stated as follows - given an iterate $x_k \in \mathbb{R}^n$, subsequent iterates are given by

$$x_{k+1} = \Pi_{\omega_k}(x_k) \tag{14}$$

where the random variable ω at time step k is a value in the set $\{1, \ldots, m\}$. Each constraint set X_i is associated with an 'agent' or 'node' that decides which set to project to next by maintaining a set of m probabilities. Each of these probabilities corresponds to projecting onto a set i. Denote this set of m probabilities associated with the node i at an iteration k as Pr so that $Pr(i, j)$ implies probability of projecting from set i to set j. Note that, traditionally in the literature, only probability of projecting onto a set $Pr(i)$ is defined. Here we can obtain this value implicitly, that is, $Pr(i) = \sum_{j=1}^{m} Pr(j, i)$.

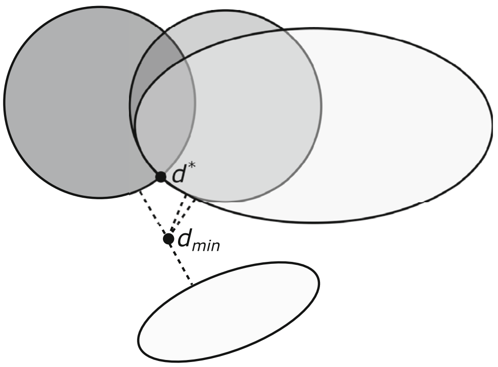

Fig. 1 We intend to find d^* at the intersection of the MFS rather than $d_{\min}$ which applies to all four sets shown in the figure

We identify the maximum feasible set (MFS) during the course of convergence by using constraint violation values. Constraint violation with respect to a constraint set X_i can be calculated, using (13), as

$$c_i = \max\left(\sum_{i \in V_{t+}} f_i(x) - f_t^*, 0\right) \tag{15}$$

Define $d_X(x)$ as the Bregman distance of a point $x \in \mathbb{R}^n$ from a set X. Notice (in Fig. 1) that a point on the intersection of the MFS does not correspond to a point with the minimum distance to all sets $d_{\min}$.

That is, we are interested in the case where:

1. $d \neq 0$, since all the sets in question (X_i's) may not intersect
2. $d \neq d_{\min}$, since some sets may be excluded from the MFS and
3. $d = d^* > d_{\min}$ which corresponds to a point on the intersection of the MFS (see Fig. 1)

Let c_{ij} be the constraint violation measured using (15) for set X_i from a point on set X_j, where $i \neq j$. We now define the adaptive probabilities $Pr(i, j)$ for iterations $k = \{0, 1, 2, \ldots\}$ in (16) :

$$Pr(i, j) = \begin{cases} 0, & \text{iff } i = j, k = 0, \\ 1 - \sum_{j \neq i} Pr(i, j), & \text{iff } i = j, k > 0, \\ \frac{2}{m-1}\left(1 - \frac{1}{1+\exp(-\gamma_k \bar{c}_{ij})}\right), & \text{iff } i \neq j, k \geq 0, \end{cases} \tag{16}$$

where $\bar{c}_{ij}$ is the moving average value of c_{ij} and γ_k is a gradually increasing sequence of positive numbers that amplify the constraint violation $\bar{c}_{ij}$ with the properties $\gamma_{k+1} > \gamma_k$, $\gamma_0 = 0$ and $\sum_{k=0}^{\infty} \gamma_k = \infty$. Pr is a doubly stochastic matrix like P that is used in the first phase (6). Note that at iteration $k = 0$, $Pr(i, j) = \frac{1}{m-1}$ for all $i \neq j$. Thus, it is equally probable to project onto other sets j from set i. As the iterations progress, the algorithm is expected to initially behave like a standard random projection algorithm to find $d_{\min}$ since γ_k is close to zero. As the average constraint

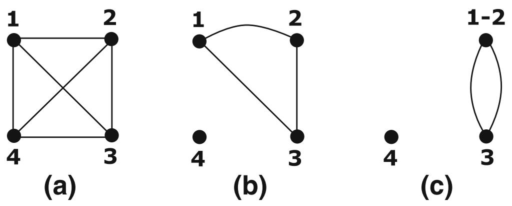

Fig. 2 **a** Initialized graph with respect to the hypothetical problem in Fig. 1. **b** Node 4 corresponding to set 4 is isolated as part of ARP. **c** Node 1 and node 2 coalesce to form node 1–2. Projections are thereafter performed on nodes 1–2 and 3

violation $\bar{c}_{ij}$ increases, it becomes less probable to project onto the set j from set i. Also, a stagnant or unchanging constraint violation is also penalized since γ_k keeps increasing. This reduces the probability of projecting onto frequently penalized nodes gradually toward zero.

An easy way to visualize the above paragraph is as follows. Let the candidate nodes ($i = \{1, \ldots, m\}$) to be added be arranged as a fully connected graph $G = (V, E)$. Initially, all links exist, and we wish to gradually fade away these links as $\bar{c}_{ij}$ increases. As the algorithm progresses, $Pr(i, j) \rightarrow 0$ implies that link does not exist. A high probability $Pr(i, j)$ and $Pr(j, i)$ indicates that sets i and j may have an intersection. On the other hand, a high value of $Pr(i, j)$ and a corresponding low probability of $Pr(j, i)$ imply that improvement of f^* is relatively higher for set corresponding to X_j than for set X_i. In this sense, ARP is an online learning algorithm. The network analogy will be referred to again since it is useful for our next step (Sect. 4.2), and also lends support to practical implementation of the algorithm on actual distributed nodes.

4.2 Continuation Via Edge Contracting

As discussed in the previous paragraph, edge contracting plays an important role in ARP. Recall that a link between two nodes implies that an intersection probably exists. Since the above procedure aims to isolate unconnected nodes while confirming pairwise intersections, we continue our analysis by using an edge contracting procedure. This is visually similar to the Karger's algorithm (See Fig. 2), but has a completely different purpose and procedure [18].

In our procedure, the result of 'contracting' an edge that connects nodes u and v is the formation of a new node $u - v$. Since $Pr(i, j)$ is not necessarily symmetric, we obtain a directed version of the graph shown in Fig. 2. At this point, we are convinced that an intersection exists between the sets X_u and X_v. Denote the set $X_{u-v} = X_u \cap X_v$, as the intersection set and the corresponding node on our graph $G = \{V, E\}$ as $u - v$. The procedure is formally described below through steps 1–5 below and begins at predetermined times following a check (if $|V| \geq 1$):

1. Choose edge $e = \{u, v\} \in E : Pr(u, v) > 0 \wedge Pr(v, u) > 0 \wedge \gamma_k \bar{c}_{uv} \rightarrow \infty$
2. Denote new node $u - v$ and perform
   (a) $G = G \setminus e$
   (b) $V = V \setminus \{u, v\}, V = V \cup u - v$
3. Isolate nodes:
   (a) Choose node $k \in V : Pr(u, k) = 0 \wedge Pr(k, u) \geq 0 \; \forall u \neq k, u \in V$

   (b) $V = V \setminus k$
4. $\gamma_{k+1} = \gamma_0$
5. Continue ARP

Practically, we look for a $Pr(u, v) > P_{\text{thr}}$ instead of $Pr(u, v) > 0$ and $Pr(u, v) < Pr_{\text{thr}}$ instead of $Pr(u, v) = 0$, where P_{thr} is some small positive threshold value. Note that a node may be isolated if all edges to it and from it are removed. After the formation of the node $u - v$, all projections are made through parallel projections onto both sets Π_{X_u} and Π_{X_v}. As a result, projections onto the set X_{u-v} may be obtained using (17):

$$\Pi_{X_{u-v}}(x) = \frac{\Pi_{X_u}(x) + \Pi_{X_v}(x)}{2} \tag{17}$$

Bauschke and Borwein [19] study the convergence of these parallel (or more generally, weighted) projections through the use of active indices in their review paper. Thus, in our algorithm we simply consider the average projection as the projection onto node $u - v$. Note that as the algorithm progresses along with edge contracting, we coalesce nodes with feasible intersections into a node denoted by $u - v - \cdots - w$. Finally, the desired MFS is obtained by the indices that identify this coalesced node. Note that in some application problems, a coalesced node may be directly associated with other nodes, and as such, it may become necessary to include these nodes into the final solution.

4.3 Mathematical Convergence of ARP

In this section, we present mathematical convergence results in relation to our algorithm. Our goal is to show that in the pivot stage,

$$\left\|x_{k+1} - x^*\right\| \leq \left\|x_k - x^*\right\| + e(x_k, \omega_k) \tag{18}$$

As in Wang and Bertsekas [20], we show this by bounding each term on the right hand side (RHS) of the above equation to show that the iteration error e, which is a function of the current iterate x_k and the random index ω_k is stochastically decreasing.

To do this, we take the conditional expectation of the 'average improvement' of the history of the process' iterates $\mathscr{F}_k = \{x_0, x_1, \ldots, x_k, \omega_0, \omega_1, \ldots, \omega_k, \gamma_0, \gamma_1, \ldots, \gamma_k\}$. That is, we analyze $E[\|x_{k+1} - x^*\|^2 \,|\mathscr{F}_k]$ where $E[\cdot]$ denotes the expected value of a random variable. Finally, we study the equivalence of two problems: one involving our version of MFS with all m sets and the second involving convergence onto a smaller number of sets which are results of solving the MFS problem a priori. As in Wang and Bertsekas [20], we assume that the collection of sets X_i satisfies linear regularity as follows.

Assumption 4.1 There exists a positive scalar η such that for any $x \in \mathbb{R}^n$,

$$\|x - \Pi(x)\| \leq \eta \cdot \max_{i \in V : G = \{V, E\}} \left\|x - \Pi_{X_i}(x)\right\| .$$

There are several situations, including polyhedral sets, where linear regularity holds true. For a detailed discussion on this property, we refer the reader to Deutsch and Hundal [21]. In simple words, the assumption roughly translates to saying that the distance of a point from two sets is closely related to the distance from the intersection of these two sets, when such an intersection exists. We also assume non-expansiveness of the projection operator Π, that is:

Assumption 4.2 For any two points x and y in $\mathbb{R}^n$, and for all projections considered

$$\|\Pi(x) - \Pi(y)\| \leq \|x - y\| .$$

Finally, also assume that we can generate an increasing sequence of numbers $\{\gamma_k\}$. We now begin our mathematical treatment of the algorithm, following the stencil provided by Wang and Bertsekas [20].

Lemma 4.1 *For any $x \in \mathbb{R}^n$, and $y \in S$ with $S \subseteq \mathbb{R}^n$,*

$$\|\Pi_S(x) - y\|^2 \leq \|x - y\|^2 - \|x - \Pi_S(x)\|^2$$

Proof Consider the quantity on the LHS, $\|\Pi_S(x) - y\|^2$

$$\begin{aligned}\|\Pi_S(x) - y\|^2 &= \|\Pi_S(x) - x + x - y\|^2 \\ &= \|x - y\|^2 + \|x - \Pi_S(x)\|^2 - 2 \cdot (x - y)'(x - \Pi_S(x)) \\ &= \|x - y\|^2 + \|x - \Pi_S(x)\|^2 \\ &\quad - 2 \cdot (x - \Pi_S(x) + \Pi_S(x) - y)'\,(x - \Pi_S(x))\end{aligned}$$

Since $(y - \Pi_S(x))'(x - \Pi_S(x)) \leq 0$

$$\begin{aligned}\|\Pi_S(x) - y\|^2 &\leq \|x - y\|^2 + \|x - \Pi_S(x)\|^2 - 2 \cdot (x - \Pi_S(x))'(x - \Pi_S(x)) \\ &= \|x - y\|^2 - \|x - \Pi_S(x)\|^2\end{aligned}$$ □

For the next Lemma, we use the following information:

1. Lemma 4.1
2. ARP iterates $x_{k+1} = \Pi_{\omega_k} x_k$
3. $2a'b \leq \epsilon\, \|a\|^2 + \frac{1}{\epsilon}\, \|b\|^2 \quad \forall a, b \in \mathbb{R}^n$ [20]
4. Distance $d_X(x) = \|x - \Pi_X(x)\|$

Lemma 4.2 *For any $x \in \mathbb{R}^n$, and $y \in S$ with $S \subseteq \mathbb{R}^n$,*

$$\|x_{k+1} - y\| \leq (1 + \epsilon)\, \|x_k - y\| + \left(1 + \frac{1}{\epsilon}\right) d^2(x_k)$$

Proof

$$\begin{aligned}\|\Pi_S(x) - y\|^2 &\leq \|x - y\|^2 + \|x - \Pi_S(x)\|^2 - 2 \cdot (x - y)'(x - \Pi_S(x)) \\ &\leq \|x - y\|^2 + \|x - \Pi_S(x)\|^2 + \left[\epsilon \|x - y\|^2 + \frac{1}{\epsilon} \|x - \Pi_S(x)\|^2\right] \\ &= (1 + \epsilon) \|x - y\|^2 + \left(1 + \frac{1}{\epsilon}\right) \cdot d^2(x)\end{aligned}$$ □

where the required result is easily obtained by substituting $x = x_k$. A consequence of this lemma can be seen by substituting $x = x_k$ and $y = x^*$, and having the set $S = X_{\omega_i}$, that is,

$$\left\|\Pi_{\omega_i}(x_k) - x^*\right\|^2 = \left\|x_{k+1} - x^*\right\|^2 \leq (1 + \epsilon) \left\|x_k - x^*\right\|^2 + \left(1 + \frac{1}{\epsilon}\right) \cdot d^2(x_k) \tag{19}$$

Although this is not of significant consequence to our 'expected' or average decrease in error, it is an interesting side note. Also, we can extend this result to an N step look-ahead as follows. We are interested to study the progress of the quantity $\|x_{k+N} - x^*\|^2$ as N grows. (19) looks one step ahead, that is, $N = 1$. For $N = 2$,

$$\begin{aligned}\left\|x_{k+2} - x^*\right\|^2 &\leq \left\|x_{k+1} - x^*\right\|^2 - d^2(x_{k+1}) \\ &\leq \left\|x_k - x^*\right\|^2 - d^2(x_k) - d^2(x_{k+1})\end{aligned} \tag{20}$$

Therefore, for an N step look-ahead we can construct (21):

$$\left\|x_{k+N} - x^*\right\|^2 \leq \left\|x_k - x^*\right\|^2 - \left[d^2(x_k) + d^2(x_{k+1}) + \cdots + d^2(x_{k+N-1})\right] \tag{21}$$

4.4 Progress Toward the Optimum

Before progressing to Proposition 4.1, we provide a lower bound of the ARP. This average progress of ARP is given as the expected value of $\left\|x - \Pi_{\omega_k}(x)\right\|$ subject to the history of the algorithm's iterates $\mathscr{F}_k$ until the kth iteration.

$$\begin{aligned}E[\left\|x - \Pi_{\omega_k}(x)\right\|^2 |\mathscr{F}_k] &= \sum_{i \in V} Pr(\omega_k = i|\mathscr{F}_k) \|x - \Pi_i(x)\|^2 \\ &= \sum_{i \in V} Pr(\omega_k = i|\omega_{k-1} = j) \|x - \Pi_i(x)\|^2 \\ &\geq \frac{2}{m-1}\left(1 - \frac{1}{1 + \exp -\gamma_k \bar{(c)}_{ij}}\right) \|x - \Pi_i(x)\|^2\end{aligned} \tag{22}$$

where $\bar{c}_i j$ corresponds to the minimum corresponding average constraint violation. Now maximizing the RHS over j and using linear regularity (see Assumption 4.1)

$$E\left[\left\|x-\Pi_{\omega_k}(x)\right\|^2|\mathscr{F}_k\right] \geq \frac{2}{\eta(m-1)}\left(1-\frac{1}{1+\exp\left(-\gamma_k\bar{c}_{ij}\right)}\right)\|x-\Pi_i(x)\|^2 \tag{23}$$

Finally, we discuss the average progress of the algorithm toward the solution x^* in the following proposition

Proposition 4.1 *Let Assumptions 4.1 and 4.2 hold, and let x^* be an optimal solution such that it converges to the MFS intersection while identifying the set itself. Then the ARP algorithm generates a sequence of iterates x_k such that*

$$E\left[d^2(x_{k+1})|\mathscr{F}_k\right] \leq \left(2+\epsilon+\frac{1}{\epsilon}\right)\cdot\frac{1}{\eta(m-1)}\cdot d^2(x_k)$$

Proof Let ϵ be a positive scalar as defined earlier. Let d represent the distance to the intersection of the MFS, and S represent the MFS. In Lemma 4.2, we use $y=\Pi_{\omega_k}x_k$ and $d^2(x_{k+1}) \leq \|x_{k+1}-\Pi_S(x_k)\|^2$ to get

$$\begin{aligned} d^2(x_{k+1}) &\leq \|x_{k+1}-\Pi_S(x_k)\|^2 \\ &\leq (1+\epsilon)\|x_k-\Pi_S(x_k)\| + \left(1+\frac{1}{\epsilon}\right)d^2(x_k) \end{aligned}$$

Now, taking the conditional expected value of both sides with respect to $\mathscr{F}_k$ and using (23), [or see (19)] we get

$$\begin{aligned} E\left[d^2(x_{k+1})|\mathscr{F}_k\right] &\leq \left(2+\epsilon+\frac{1}{\epsilon}\right)\cdot\frac{2}{\eta(m-1)}\cdot\left(1-\frac{1}{1+\exp\left(-\gamma_k\bar{c}_{ij}\right)}\right)\cdot d^2(x_k) \\ &\leq \left(2+\epsilon+\frac{1}{\epsilon}\right)\cdot\frac{1}{\eta(m-1)}\cdot d^2(x_k) \end{aligned}$$ □

4.5 Implication of γ_k Sequence

The purpose of γ_k is to gradually amplify the effect of the average constraint violation $\bar{c}_{ij}$. For the purpose of this discussion, let us assume that the sequence of $\{\gamma_k\}$ is generated using:

$$\gamma_{k+1} = r_\gamma \cdot \gamma_k \tag{24}$$

with $\gamma_0 > 0$, and rate of increase $r_\gamma > 0$. It is obvious that the rate r_γ must be upper-bounded to avoid artificial or premature convergence. This may lead to a wrong choice of nodes that form the MFS, although some node that improves the value of f^* may be found. However, the value of this upper bound depends on the rate of variation of the average constraint violation $\bar{c}_{ij}$ which is problem specific.

5 Application to a Sensor Management Problem

There has been a growing interest in topics related to distributed optimization studies of sensor management applications (see [22], for example). Generally, optimization involves improving the value of some system performance metric such as information maximization or risk minimization. We include an application problem in the sensor management domain due to typical problem features that make application of our algorithm to this problem very apt such as (1) distributed and localized function evaluations, (2) optimization of a global goal by using local interactions, (3) the need to add additional sensors/nodes/targets to the network, each of which is typically associated with a local function and (4) the possibility of onboard computing.

Robust estimation is a popularly solved subproblem in this category (see [5,23–30]). Each sensor in a network may collect local readings of an environmental variable (like temperature or rainfall), or of a location parameter (like an energy source or target), subject to noise. Robust estimates of parameters and locations are often obtained from functions such as squared error and the Huber loss function [25]. Here, we will make use of the latter function as the local f_i being calculated by each sensor. The Huber loss function is given by:

$$\rho(\theta, x) = \begin{cases} \frac{(x-\theta)^2}{2}, & \text{if } |x-\theta| \le \gamma\,, \\ \gamma|x-\theta| - \frac{\gamma^2}{2}, & \text{otherwise,} \end{cases} . \tag{25}$$

with the multi-dimensional variant involving the sum of these losses across all dimensions. The Huber loss function is convex and differentiable. A general description of the robust source estimation follows. Each sensor $i = \{1, \ldots, N\}$ collects a set of m measurements x_i which are randomly sampled from normal distributions $\mathcal{N}(\theta, \sigma^2)$. 75 % of the sensors sample the source location θ with a noise characterized by σ^2, whereas the other 25 % are faulty sensors with a noise of $10\sigma^2$. We would like to find an estimate of the source $\theta \in X$ which minimizes :

$$\begin{gathered} f(\theta, x) = \frac{1}{n}\sum_{i=1}^{n} \rho_i(\theta, x) \\ x \in X \end{gathered} \tag{26}$$

In our problem, we randomly generate a network with $N = 100$ nodes uniformly distributed on the unit square $[0, 1] \times [0, 1]$ (X) as shown in Fig. 3a. We then connect the nodes based on whether their distance is <0.145 units (obtained by reducing the threshold value until the network remains connected) similar to the example in [31]. In the context of our paper, any new sensors added will also follow the same rule of establishing connections with other existing nodes. This relates to the real-world situation where wireless sensors may connect to other sensors within some range. Each sensor processes $m = 200$ readings. The location of the sensors and the source are randomly generated. A set of 16 new candidate sensors (positions shown in Fig. 3b) are given access to the last 50 readings of the original set of 100 sensors.

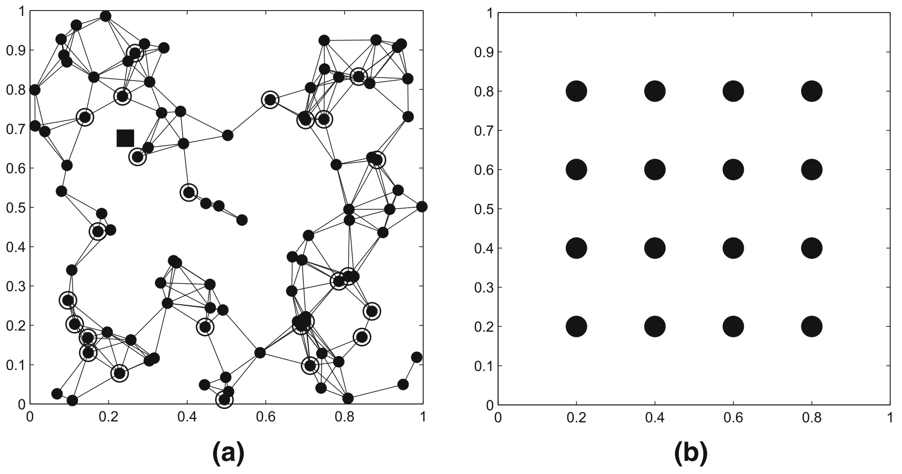

Fig. 3 In both images shown above, *filled circle* represents a sensor, *filled square* represents the source's location, and a *open circle* around a sensor indicates that it is faulty (also randomly generated). **a** Randomly generated initial network. **b** New candidate sensors

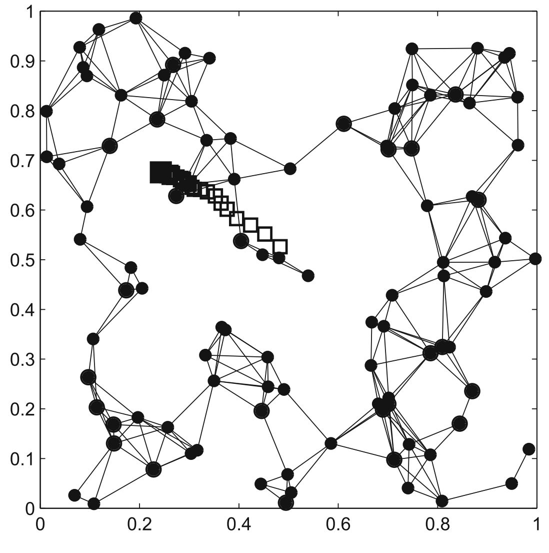

Fig. 4 Mean estimate of the source improving with number of iterations

We first solve the problem with the original set of 100 sensors in the randomly generated network using DDA. The aim is then to use the best source location obtained as a result of the DDA algorithm to improve f^* value while simultaneously selecting the most valuable, maximum subset of the 16 new candidate sensors.

5.1 Results

A custom MATLAB 2014a code running on an Intel Core i7-2630QM 2GHz processor was used. The sequence of γ_k was generated as a simple geometric sequence with $\gamma_0 = 1$ and $\gamma_{k+1} = \gamma_k \cdot 1.0015$ (refer Sect. 4.5 for discussion on rate of increase of γ).

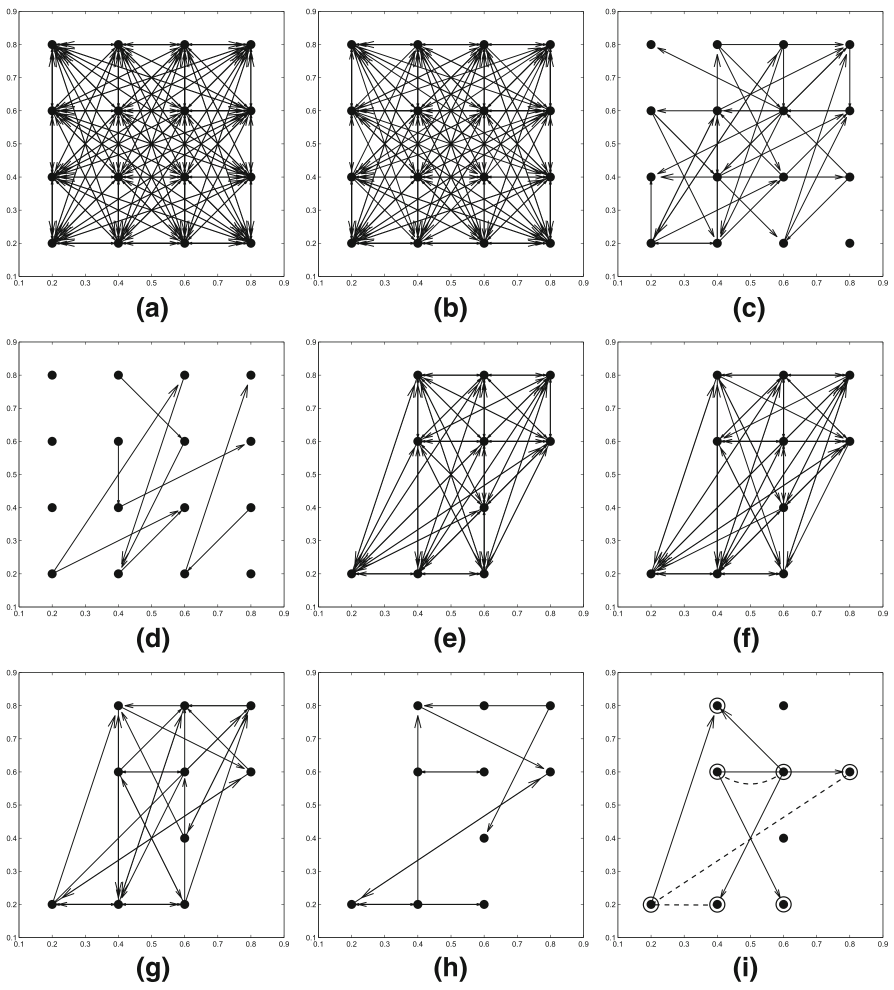

Fig. 5 Progress of the ARP algorithm for selection of a subset of the 16 candidate sensors shown as edge contraction of a directed graph corresponding to $Pr(i, j)$ values after the kth iteration number. *Circled nodes* are selected, and *dashed lines* connect coalesced nodes. **a** $k = 1000$, **b** $k = 2000$, **c** $k = 3000$, **d** $k = 4000$, **e** $k = 5000$, **f** $k = 5200$, **g** $k = 5400$, **h** $k = 5600$, **i** $k = 5821$

The value of α for the initial run of the DDA algorithm is fixed at a value of 0.01. The path taken by the mean source location is shown in Fig. 4.

As we can see, the mean estimate of the source location continuously improves toward the actual location as the DDA algorithm progresses. Figure 4 shows the path taken by the mean estimate of the source (shown as open square) moving toward the actual source location filled square as time progresses in the presence of faults. Sizes of the sensors filled circle have been reduced for clarity.

The maximum number of projections in ARP is set at 5000 (implying usability in on-board systems) Also, for coalescing nodes, we use a threshold probability value of $Pr_{\text{thr}} = 0.05$. Corresponding to the $Pr(i, j)$ values, we obtain characteristic directed

Table 1 Comparison of the four test cases in terms of number of mean, standard deviation, best and worst f values (iterations are not reported since they are related to the number of readings taken, which in this case is always 250)

Case	Mean f	SD f	Best f	Worst f
Case a	0.1376	0.2759	1.356e−4	1.020
Case b	0.1146	0.2840	9.839e−5	1.654
Case c	0.1135	0.2329	1.556e−4	1.264
Case d	0.1363	0.2719	2.191e−4	1.017

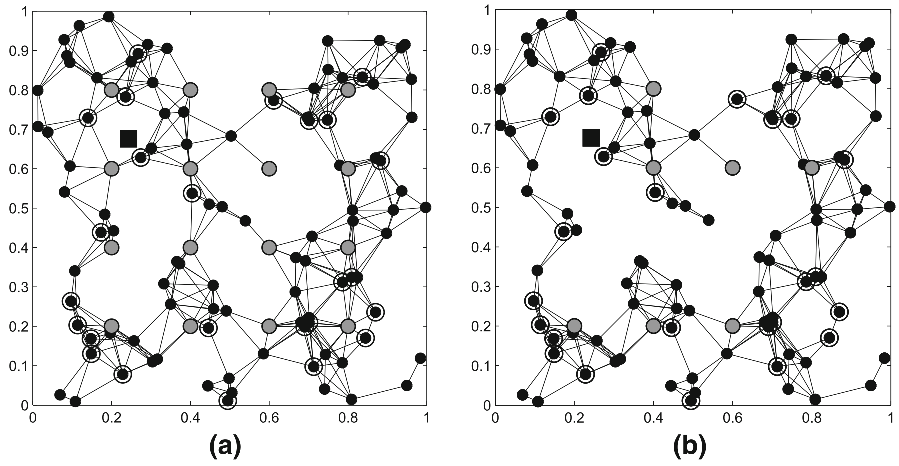

Fig. 6 Refer to cases in text : **a** Case b: final network after ARP selection. **b** Case c: network used when all candidate sensors are added a priori. New sensors are *colored gray*

graphs for the purpose of edge contraction and isolation as shown in Fig. 5. ARP restarts after edge contraction and isolation procedures take place (Note transition between Figs. 5d, e). At $k = 5600$ (Fig. 5h), four pairs of nodes (seen with bidirectional links) coalesce to form new nodes while allowing the corresponding edges to contract. In Fig. 5i we see the result of ARP with a final converged set of nodes. A clear improvement in f^* is noticed while selecting candidate sensors (see Table 1).

We now compare these results (mean f value) with a centralized optimization counterpart with the following cases—Case (a) Original 100 node problem, Case (b) All 16 candidate nodes are added to the network (Fig. 6a), Case (c) Nodes selected by ARP are added a priori (Fig. 6b) and Case (d) Solved using DDA and ARP. Since the mean f value itself oscillates at each iteration in this example, the best and worst f values are also reported.

It is important to note that Cases a, b and c are run with varying numbers of sensors, and therefore sample and distribute information differently. Furthermore, these cases are run for exactly 200 time steps (or iterations of the DDA algorithm). Case d, on the other hand, includes the pivot phase of the ARP algorithm for sensor selection and improvement. The values reported in columns of Table 1 are valid at the 200th iteration. It is possible for other iterations to portray different comparisons since the problem is stochastic in nature. That being said, the ARP algorithm selects useful sensors and improves f^*. These additional 7 sensors, when added to the network a priori (Case c, Fig. 6b) and solved using DDA, report the best mean f value. When all 16 sensors are added (Case b, Fig. 6a), the mean, best and worst values are slightly inferior to Case c, although this may be an artifact of the aforementioned stochastic nature of the problem.

6 Conclusions

An evolving counterpart of the standard minimum sum problem involving convex functions is presented. We introduce a pivot phase in which new nodes (corresponding to new functions) are selected by first converting the overall optimization problem to a convex feasibility problem and then deriving the MFS. Proof of convergence of this adaptive random projection (ARP) algorithm is provided. A relation for the upper bound of the rate of increase in the sequence $\{\gamma_k\}$ is an important area that is yet to be explored. We solve an application problem which involves robust source localization in the setting of evolving design spaces. Evolving the design space and solving using ARP allows us to improve the design by intelligently reusing previously obtained results. Comparisons were also made with alternate cases where nodes or agents selected by the ARP algorithm are added a priori, and also when all candidate nodes are added at once. The algorithm is widely applicable and may be important in problems where the subset of candidate nodes or agents to be added for an overall improvement is completely unknown.

References

1. Bertsekas, D.P.: Incremental gradient, subgradient, and proximal methods for convex optimization: a survey. Optim. Mach. Learn. **2010**, 1–38 (2011)
2. Bertsekas, D.P.: Incremental proximal methods for large scale convex optimization. Math. Program. **129**(2), 163–195 (2011)
3. Nedic, A.: Random projection algorithms for convex set intersection problems. In: 2010 49th IEEE Conference on Decision and Control (CDC), pp. 7655–7660. IEEE (2010)
4. Nedic, A., Ozdaglar, A.: Distributed subgradient methods for multi-agent optimization. IEEE Trans. Autom. Control **54**(1), 48–61 (2009)
5. Rabbat, M.G., Nowak, R.D.: Quantized incremental algorithms for distributed optimization. IEEE J. Sel. Areas Commun. **23**(4), 798–808 (2005)
6. Duchi, J.C., Agarwal, A., Wainwright, M.J.: Dual averaging for distributed optimization: convergence analysis and network scaling. IEEE Trans. Autom. Control **57**(3), 592–606 (2012)
7. Johansson, B., Rabi, M., Johansson, M.: A randomized incremental subgradient method for distributed optimization in networked systems. SIAM J. Optim. **20**(3), 1157–1170 (2009)
8. Ram, S.S., Nedić, A., Veeravalli, V.V.: Distributed stochastic subgradient projection algorithms for convex optimization. J. Optim. Theory Appl. **147**(3), 516–545 (2010)
9. Censor, Y., De Pierro, A.R., Zaknoon, M.: Steered sequential projections for the inconsistent convex feasibility problem. Nonlinear Anal. Theory Methods Appl. **59**(3), 385–405 (2004)
10. De Pierro, A.R., Iusem, A.: A relaxed version of Bregman's method for convex programming. J. Optim. Theory Appl. **51**(3), 421–440 (1986)
11. Censor, Y., Lent, A.: An iterative row-action method for interval convex programming. J. Optim. Theory Appl. **34**(3), 321–353 (1981)

12. Censor, Y., Motova, A., Segal, A.: Perturbed projections and subgradient projections for the multiple-sets split feasibility problem. J. Math. Anal. Appl. **327**(2), 1244–1256 (2007)
13. Byrne, C.: Bregman-Legendre multidistance projection algorithms for convex feasibility and optimization. Stud. Comput. Math. **8**, 87–99 (2001)
14. Aharoni, R., Berman, A., Censor, Y.: An interior points algorithm for the convex feasibility problem. Adv. Appl. Math. **4**(4), 479–489 (1983)
15. Chinneck, J.W.: Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods, vol. 118. Springer, Berlin (2007)
16. Amaldi, E., Pfetsch Jr., M.E., Trotter L.E.: Some structural and algorithmic properties of the maximum feasible subsystem problem. In: Integer Programming and Combinatorial Optimization, pp. 45–59. Springer (1999)
17. Pfetsch, M.E.: Branch-and-Cut for the Maximum Feasible Subsystem Problem. Konrad-Zuse-Zentrum fr Informationstechnik, Berlin (2005)
18. Karger, D.R.: Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In: Proceedings of 4th Annual ACM-SIAM Symposium on Discrete Algorithms, vol. 93 (1993)
19. Bauschke, H.H., Borwein, J.M.: On projection algorithms for solving convex feasibility problems. SIAM Rev. **38**(3), 367–426 (1996)
20. Wang, M., Bertsekas, D.P.: Incremental constraint projection-proximal methods for nonsmooth convex optimization. Technical report, MIT (2013)
21. Deutsch, F., Hundal, H.: The rate of convergence for the cyclic projections algorithm III: regularity of convex sets. J. Approx. Theory **155**(2), 155–184 (2008)
22. Hero, A.O., Cochran, D.: Sensor management: past, present, and future. Sens. J., IEEE **11**(12), 3064–3075 (2011)
23. Cattivelli, F.S., Lopes, C.G., Sayed, A.H.: Diffusion recursive least-squares for distributed estimation over adaptive networks. IEEE Trans. Signal Process. **56**(5), 1865–1877 (2008)
24. Kekatos, V., Giannakis, G.B.: Distributed robust power system state estimation. IEEE Trans. Power Syst. **28**(2), 1617–1626 (2013)
25. Rabbat, M., Nowak, R.: Distributed optimization in sensor networks. In: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, pp. 20–27. ACM (2004)
26. Huber, P.J., et al.: Robust estimation of a location parameter. Ann. Math. Stat. **35**(1), 73–101 (1964)
27. Léger, J.B., Kieffer, M.: Guaranteed robust distributed estimation in a network of sensors. In: 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 3378–3381. IEEE (2010)
28. Delouille, V., Neelamani, R., Baraniuk, R.: Robust distributed estimation in sensor networks using the embedded polygons algorithm. In: Proceedings of the 3rd International Symposium on Information processing in Sensor Networks, pp. 405–413. ACM (2004)
29. Moore, D., Leonard, J., Rus, D., Teller, S.: Robust distributed network localization with noisy range measurements. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pp. 50–61. ACM (2004)
30. Li, Q., Wong, W.: Optimal estimator for distributed anonymous observers. J. Optim. Theory Appl. **140**(1), 55–75 (2009)
31. Xiao, L., Boyd, S., Lall, S.: A scheme for robust distributed sensor fusion based on average consensus. In: Fourth International Symposium on Information Processing in Sensor Networks, 2005. IPSN 2005, pp. 63–70. IEEE (2005)